## REMARKS

The Office Action of May 23, 2006 has been received and carefully considered. All claims are now present for examination and favorable reconsideration is respectfully requested in view of the following comments.

REJECTIONS UNDER 35 U.S.C. § 102:

Claims 1, 3 and 5 have been rejected under 35 U.S.C. § 102 (b) as allegedly being anticipated by Schneier (Bruce Schneier, Applied Cryptography, 1996, John Wiley & Sons). The Examiner indicated that "The right half of the data is expanded to 48 bits via an expansion permutation, combined with 48 bits of a shifted and permuted key via an Xor ...". However, having carefully reviewed the Office Action and analyzed the description of algorithm DES in Schneier in detail, Applicant respectfully submits that, Schneier, when describing algorithm DES, does not disclose any feature of converting a subkey depending on data being converted. The Examiner has incorrectly interpreted the conversion operation $f$ appeared on page 270 and in Fig. 12.1 as an operation of permuting bits of subkey $K_I$. In fact, the conversion operation $f$ is not a permutation operation. This is supported by the following evidence.

The number of unit bits in a binary vector produced as a result of combined conversion of subkey $K_I$ and data subblock $R_0$ (page 270 and Fig. 12.1) is typically **not equal to** the number of unit and zero bits in either subkey $K_I$ or in data subblock $R_0$. However, as a result of performing the operation of bit permutation, the number of unit bits in the output binary vector **is always equal to** the number of unit bits in the input binary vector. This is because, in performing the operation of permutation, the bits are permuted but not inverted. Only with a very small probability, as a result of performing the conversion operation $f$, may the number of unit bits at the output by chance coincide with the number of unit bits in subkey $K_I$ or in data subblock $R_0$. The fact that the conversion operation $f$ is not an operation of permutation, is also supported by B.

Schneier, A. Menezes (see Menezes A.J., Vanstone S.A., Handbook of Applied Cryptography, CRC Press, 1996. 780p.) and J. Buchmann (Buchmann J. Introduction to Cryptography, Springer-Verlag, New York, Berlin, 2004. 335p.) Thus, the conversion operation *f* is not an operation of bit permutation and, hence, is not an operation of bit permutation performed depending on the data subblock being converted. Therefore, algorithm DES does not anticipate the claimed method of block encryption.

It is respectfully submitted that the interpretation of the conversion operation *f* as provided by Applicant in the previous response to the office action is correct and corresponds to the description in Schneier. Schneier clearly does not stress that the conversion operation *f* consists in performing several consecutive elementary operations. In Fig. 12.1, Schneier presents a general scheme of conversions of algorithm DES as a sequence of performing sixteen typical conversion rounds and designates one conversion round as the conversion operation *f*. Then, in Fig. 12.2, Schneier discloses elementary operations constituting each round. Schneier detailed elementary operations constituting the conversion operation *f* in the sections immediately following page 270 and Fig. 12.1. These sections are "The Expansion Permutation" (page 273), "The S-Box Substitution" (pages 274-275), "The P-Box Permutation" (pages 275-277). Because these sections immediately follow the description of the general scheme of the make-up of algorithm, Schneier clearly did not indicate that these sections described consecutive elementary operations, which specify the conversion operation *f*. Apparently, this circumstance misled the Examiner. The fact that the conversion operation *f* in algorithm DES is a composite operation and includes a sequence of elementary conversion operations, follows from the description of DES provide also in the widely known document -- Menezes (see Menezes A.J., Vanstone S.A., Handbook of Applied Cryptography, CRC Press, 1996. 780p.), on page 255 (Figs. 7.10)(copy enclosed). The document Menezes clearly discloses that peforming the conversion operation *f* is a sequence of the following elementary operations: i) subblock $R_{i-1}$ expansion operation (e.g. of subblock $R_0$), ii) modulo 2 bit-by-bit summation operation performed on an expanded data subblock and subkey $K_i$ (e.g. subkey $K_1$), iii) substitution operation performed on the result obtained at the output of the preceding operation, and iv) fixed bit permutation operation. Here, the fixed bit permutation operation does not depend on any data subblock being converted. The same fixed bit permutation is performed for all possible input data in a given round when performing the operation *f* at step iv). The document J. Buchmann (Buchmann J. Introduction to Cryptography, Springer-Verlag, New York, Berlin, 2004. 335p.) also indicates that the conversion step operation *f* used in algorithm DES consists of the above elementary operations i), ii), iii) and iv), on page 131 (copy enclosed). Therefore, the conversion operation *f* is not a bit permutation operation and does not include any bit permutation operation dependent on a data subblock. Accordingly, algorithm DES does not anticipate the claimed invention.

In addition, although page 270 of Schneier indicates that "The right half of the data is expanded to 48 bits via an expansion permutation, combined with 48 bits of a shifted and permuted key via an Xor ...", one of ordinary skill in the art cannot conclude from this reference that a bit permutation operation was previously the subkey **depending on** some **data subblock** being converted. On page 272 and in section "The Key Transformation", Schneier discloses, what bit permutation operation was performed on the subkey: "First, the 56-bit key is divided into two 28-bit halves. Then, the halves are circularly shifted left by either or two bits, depending on the round." Thus, in algorithm DES, the **bit permutation operation** is performed on the key by **depending on the number of the round, but not on the data subblock, i.e. the feature of performing the subkey bit permutation operation depending on the data subblock being converted**, that is presented in the claimed invention. Furthermore, the newly presented Claim 5 include additional feature of value of another subkey, which is not disclosed or suggested in prior art, including Schneier.

Therefore, the newly presented claim is not anticipated by Schneier and the rejection under 35 U.S.C. § 102 (b) has been overcome. Accordingly, withdrawal of the rejection under 35 U.S.C. § 102 (b) is respectfully requested.

Having overcome all outstanding grounds of rejection, the application is now in condition for allowance, and prompt action toward that end is respectfully solicited.

Respectfully submitted,

JACOBSON HOLMAN PLLC

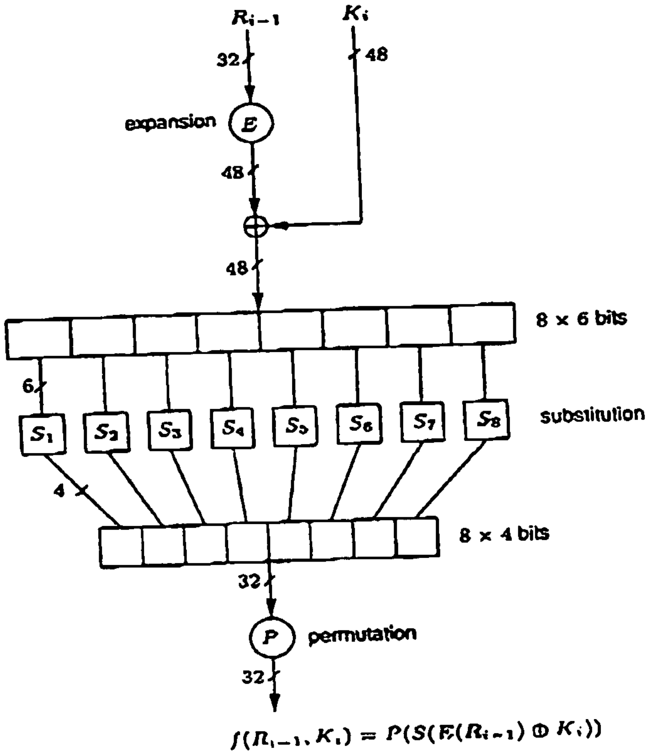
By [illegible]
John C. Holman
Registration No. 22,769

Date: August 22, 2006
(202) 638-6666
400 Seventh Street, N.W.
Washington, D.C. 20004
Atty. Dkt. No.: P65855US0

Enclosures:

1. Copy of Page 255 of Menezes A.J., Vanstone S.A., Handbook of Applied Cryptography, CRC Press, 1996. 780pp.

2. Copy of Page 131 of Buchmann J. Introduction to Cryptography, Springer-Verlag, New York, Berlin, 2004. 335pp.



$R_{i-1}$
$K_i$
32
48
expansion
$E$
48
48
8 × 6 bits
6
$S_1$ $S_2$ $S_3$ $S_4$ $S_5$ $S_6$ $S_7$ $S_8$
substitution
4
8 × 4 bits
32
$P$
permutation
32
$f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$

*Figure 7.10: DES inner function f.*

**7.83 Algorithm** DES key schedule

INPUT: 64-bit key $K = k_1 \dots k_{64}$ (including 8 odd-parity bits).
OUTPUT: sixteen 48-bit keys $K_i$, $1 \le i \le 16$.

1. Define $v_i$, $1 \le i \le 16$ as follows: $v_i = 1$ for $i \in \{1, 2, 9, 16\}$; $v_i = 2$ otherwise. (These are left-shift values for 28-bit circular rotations below.)
2. $T \leftarrow \text{PC1}(K)$; represent $T$ as 28-bit halves $(C_0, D_0)$. (Use PC1 in Table 7.4 to select bits from $K$: $C_0 = k_{57}k_{49} \dots k_{36}$, $D_0 = k_{63}k_{55} \dots k_4$.)
3. For $i$ from 1 to 16, compute $K_i$ as follows: $C_i \leftarrow (C_{i-1} \hookleftarrow v_i)$, $D_i \leftarrow (D_{i-1} \hookleftarrow v_i)$, $K_i \leftarrow \text{PC2}(C_i, D_i)$. (Use PC2 in Table 7.4 to select 48 bits from the concatenation $b_1b_2 \dots b_{56}$ of $C_i$ and $D_i$: $K_i = b_{14}b_{17} \dots b_{32}$. '$\hookleftarrow$' denotes left circular shift.)

If decryption is designed as a simple variation of the encryption function, savings result in hardware or software code size. DES achieves this as outlined in Note 7.84.

**7.84 Note** (*DES decryption*) DES decryption consists of the encryption algorithm with the same key but reversed key schedule, using in order $K_{16}, K_{15}, \dots, K_1$ (see Note 7.85). This works as follows (refer to Figure 7.9). The effect of $\text{IP}^{-1}$ is cancelled by IP in decryption, leaving $(R_{16}, L_{16})$; consider applying round 1 to this input. The operation on the left half yields, rather than $L_0 \oplus f(R_0, K_1)$, now $R_{16} \oplus f(L_{16}, K_{16})$ which, since $L_{16} = R_{15}$ and $R_{16} = L_{15} \oplus f(R_{15}, K_{16})$, is equal to $L_{15} \oplus f(R_{15}, K_{16}) \oplus f(R_{15}, K_{16}) = L_{15}$. Thus round 1 decryption yields $(R_{15}, L_{15})$, i.e., inverting round 16. Note that the cancellation

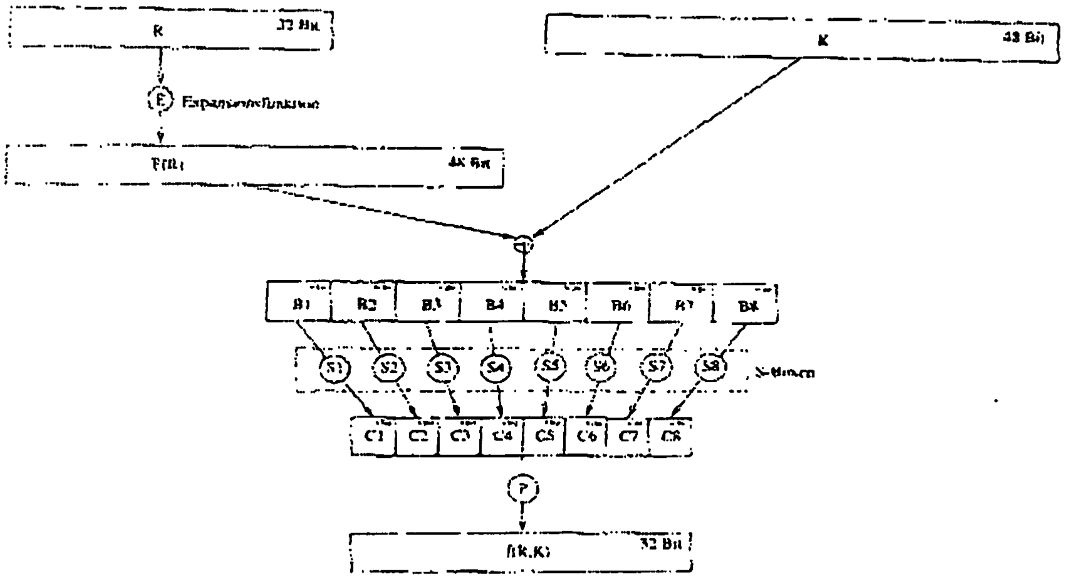


**FIGURE 5.1** The $f$-function of DES.

is computed with $B_i \in \{0,1\}^6$, $1 \le i \le 8$. In the next step, functions

$$S_i : \{0,1\}^6 \to \{0,1\}^4, \quad 1 \le i \le 8$$

are used (the so-called $S$-boxes). They are described below. Using those functions, the string

$$C = C_1C_2C_3C_4C_5C_6C_7C_8$$

is determined, where $C_i = S_i(B_i)$, $1 \le i \le 8$. It has length 32. The permutation $P$ from Table 5.3 is applied to this string. The result is $f_K(R)$.

## 5.2.4 $S$-boxes

Now we describe the $S$-boxes $S_i$, $1 \le i \le 8$. They are the heart of DES because they are highly nonlinear (see Exercise 5.5.6). They are shown in Table 5.4.Each $S$-box is represented by a table with four rows and 16 columns. For each string $B = b_1b_2b_3b_4b_5b_6$, the value $S_i(B)$ is computed as follows. The integer with binary expansion $b_1b_6$ is used as the row index. The integer with binary expansion $b_2b_3b_4b_5$ is used as the column index. The entry of the $S$-box in this row and column is written in binary expansion. This expansion is padded with leading zeros such that its length is four. The result is $S_i(B)$.